

CLIPPEDIMAGE= JP409330230A

PAT-NO: JP409330230A

DOCUMENT-IDENTIFIER: JP 09330230 A

TITLE: METHOD AND SYSTEM FOR TRANSPLANTING APPLICATION  
BINARY INTERFACE

PUBN-DATE: December 22, 1997

INVENTOR-INFORMATION:

NAME

SCOTT, HARRISON DANFORCE

PURAKASHIYU, BUINODORAI DESAI

ASSIGNEE-INFORMATION:

NAME

COUNTRY

INTERNATL BUSINESS MACH CORP <IBM>

N/A

APPL-NO: JP09054658

APPL-DATE: March 10, 1997

INT-CL\_(IPC): G06F009/45

ABSTRACT:

PROBLEM TO BE SOLVED: To provide an information processing system equipped with an application programming interface(API) with which an application binary interface(ABI) can be transplanted from a certain platform to another non-compatible platform almost without difficulty.

SOLUTION: Inside the information processing system, there are one or plural processors, storage system, one or plural input/output controllers, system bus for connecting the processors, storage system and input/output controllers, and operating system programmed to control the operation of the information processing system. The API is operated while being interlocked with an operating program. The API provides an interface neutral for languages and platforms for operating a specified parameter list inside the ABI and can be

transplanted from a certain platform to the other platform.

COPYRIGHT: (C)1997,JPO

(19) 日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-330230

(43) 公開日 平成9年(1997)12月22日

(51) Int. Cl. <sup>4</sup>	識別記号	庁内整理番号	FI	技術表示箇所
G 0 6 F 9/45			G 0 6 F 9/44	3 2 0 A

審査請求 未請求 請求項の数21 O L (全 10 頁)

(21) 出願番号 特願平9-54658  
 (22) 出願日 平成9年(1997)3月10日  
 (31) 優先権主張番号 08/619061  
 (32) 優先日 1996年3月20日  
 (33) 優先権主張国 米国 (US)

(71) 出願人 390006531  
 インターナショナル・ビジネス・マシーンズ・コーポレーション  
 INTERNATIONAL BUSINESS MACHINES CORPORATION  
 アメリカ合衆国10504、ニューヨーク州アーモンク (番地なし)  
 (72) 発明者 スコット・ハリソン・ダンフォース  
 アメリカ合衆国78750 テキサス州オースチン ウッドランド・ビレッジ 10011  
 (74) 代理人 弁理士 合田 潔 (外2名)

最終頁に続く

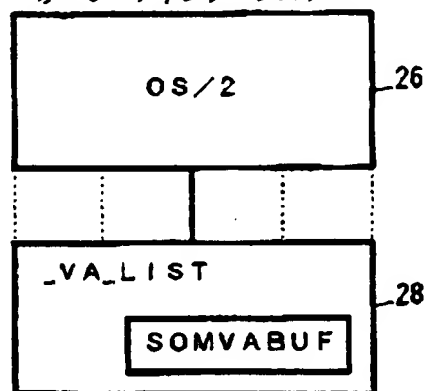
(54) 【発明の名称】 アプリケーション・バイナリ・インターフェースを移植する方法及びシステム

(57) 【要約】

【課題】 アプリケーション・バイナリ・インターフェースを、あるプラットフォームから別の非互換プラットフォームにほとんど困難なしに移植することができるアプリケーション・プログラミング・インターフェース (API) を含む情報処理システムを提供する。

【解決手段】 情報処理システム内には、1つまたは複数のプロセッサ、記憶システム、1つまたは複数の入出力制御装置、プロセッサと記憶システムと入出力制御装置を接続するシステム・バス、及び情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムがある。APIは、オペレーティング・プログラムと連動して動作する。APIは、ABI内の特定の変数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供し、あるプラットフォームから別のプラットフォームに移植可能である。

オペレーティング・システム



(2)

特開平9-330230

1

## 【特許請求の範囲】

【請求項1】1つまたは複数のプロセッサと、

記憶システムと、

1つまたは複数の人出力制御部と、

プロセッサ、記憶システム及び入出力制御装置を接続するシステム・バスと、

情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムと、

オペレーティング・システム・プログラムと連動して動作し、変数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース（API）プログラムを含む情報処理システム。

【請求項2】アプリケーション・プログラミング・インターフェース（API）プログラムが、さらに、ユーザの可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする請求項1に記載のシステム。

【請求項3】APIプログラムがさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を加える手段を含むことを特徴とする請求項2に記載のシステム。

【請求項4】APIプログラムがさらに、前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする請求項3に記載のシステム。

【請求項5】APIプログラムがさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする請求項3に記載のシステム。

【請求項6】前記APIがさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す手段を含むことを特徴とする請求項3に記載のシステム。

【請求項7】前記APIがさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第1引数を入れることを可能にする手段を含むことを特徴とする請求項3に記載のシステム。

【請求項8】変数リストを操作して、アプリケーション・バイナリ・インターフェースを所与のプラットフォームから1つまたは複数の追加プラットフォームに移植するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース（API）を提供する手段を含むコンピュータ読取り媒体。

【請求項9】アプリケーション・プログラミング・インターフェースがさらに、

ユーザ可変引数を含むこともある選択されたオブジェ

2

クト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする請求項8に記載のコンピュータ読取り媒体。

【請求項10】API手段がさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を追加する手段を含むことを特徴とする請求項9に記載のコンピュータ読取り媒体。

【請求項11】API手段がさらに、前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする請求項9に記載のコンピュータ読取り媒体。

【請求項12】API手段がさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする請求項8に記載のコンピュータ読取り媒体。

【請求項13】API手段がさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す手段を含むことを特徴とする請求項8に記載のコンピュータ読取り媒体。

【請求項14】API手段がさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第1引数を入れることを可能にする手段を含むことを特徴とする請求項8に記載のコンピュータ読取り媒体。

【請求項15】アプリケーション・バイナリ・インターフェース（ABI）を、情報処理システム上で動作する選択されたプラットフォームから1つまたは複数の追加プラットフォームに移植する方法であって、変数リストを操作して、前記ABIを前記選択されたプラットフォームから前記1つまたは複数の追加プラットフォームに移植するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェースを提供する段階を含む方法。

【請求項16】前記アプリケーション・プログラミング・インターフェース（API）がさらに、ユーザ可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する段階を含むことを特徴とする請求項15に記載のABIを移植する方法。

【請求項17】前記APIがさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を追加する段階を含むことを特徴とする請求項16に記載のABIを移植する方法。

【請求項18】前記APIがさらに、前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する段階

(3)

特開平9-330230

3

を含むことを特徴とする請求項16に記載のABIを移植する方法。

【請求項19】前記APIがさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する段階を含むことを特徴とする請求項16に記載のABIを移植する方法。

【請求項20】前記APIがさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラー値を取り出す段階を含むことを特徴とする請求項16に記載のABIを移植する方法。

【請求項21】前記APIがさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リスト内に最初の引数を入れることを可能にする段階を含むことを特徴とする請求項16に記載のABIを移植する方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般に複数のプラットフォームを支援する情報処理システムに関する。より具体的には、本発明は、処理システム内で動作するプラットフォーム間で言語及びプラットフォームに中立なインターフェース処理を提供するアプリケーション・プログラミング・インターフェースに関する。

【0002】

【従来の技術】電子計算システムは、仕事や私生活のほとんどあらゆる側面に浸透している。コンピュータ・システムは、仕事面では、大量のデータの管理やそのようなデータの処理に使用される。このようなデータ処理が必要なため、様々なコンピュータ・システム上で動作するいくつかのプラットフォームが提供されている。このようなプラットフォームは、通常、各社独自のものであり、多くの場合ソフトウェア変換のために相互に対話するのが困難である。それが重要なのは、あるプラットフォーム上で強い人気をもつ特定のソフトウェア・プログラムを別のプラットフォームに移植するとき、プラットフォームの言語に互換性がないために、そのような人気のあるソフトウェアを新しいプラットフォームに簡単に移植することができないからである。

【0003】それぞれ各プラットフォームは、固有のアプリケーション・バイナリ・インターフェース(ABI)を有し、それが、メモリ内に可変引数がどのように記憶されているかを決定する。C言語では、Cライブラリ内に、可変数の引数を入力として受け取ると予想される機能のスタック・フレーム上に可変引数リストを作成するインターフェースが提供される。この場合、残念ながら、可変引数リストの作成を必要とするソフトウェア開発業者は、リンケージ規約の詳細と可変引数が所与のプラットフォームに対してどのように表されるかを知ら

4

なければならない。様々なプラットフォームが異なるABIを有する場合、複数のプラットフォーム用の移植可能コードを記述するタスクはほとんど不可能になる。

【0004】

【発明が解決しようとする課題】したがって、アプリケーションを複数のプラットフォーム間で大した困難なしに移植できるようにするAPIが必要である。

【0005】

【課題を解決するための手段】本発明によれば、アプリケーション・バイナリ・インターフェースをあるプラットフォームから別の非互換プラットフォームにほとんど困難なしに移植できるように改良されたアプリケーション・プログラミング・インターフェースを含む情報処理システムが開示される。情報処理システム内には、1つまたは複数のプロセッサ、記憶システム、1つまたは複数の入出力制御装置、プロセッサと記憶システムと入出力制御装置を接続するシステム・バス、及び情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムがある。アプリケーション・プログラミング・インターフェース(API)は、オペレーティング・プログラムと連動して動作する。APIは、ABI内の特定の可変引数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供し、あるプラットフォームから別のプラットフォームに移植可能である。

【0006】アプリケーション・プログラミング・インターフェースは、さらに、選択されたオブジェクト変数バッファ構造を作成し初期設定する段階を含む。オブジェクト変数バッファ構造は、ユーザ可変引数を含むこともある。さらに、APIは、オブジェクト変数バッファ構造の1つに含まれる変数リストに渡された引数を追加する段階を含む。APIは、また、オブジェクト変数バッファ構造内の可変引数リストを入力引数として渡された変数リストに複写する手段も含む。さらに、APIは、オブジェクト変数バッファ構造と、オブジェクト変数バッファ構造内に含まれる変数リストとに関連付けられた任意の割り振りメモリを消去する段階を含む。さらに、APIは、オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラー値を取り出す手段を含む。また、APIはさらに、ユーザが、オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに最初の引数を入れることを可能にする手段を含む。APIは、情報処理システム内にロードされたプログラムまたはコンピュータ読取り媒体上に記憶されたプログラムとして実施され、あるいはプログラムを特定のプラットフォームから第2の非互換プラットフォームに移植する方法として実施されることが開示される。

【0007】

【発明の実施の形態】次に、図1と図2を参照し、本発明の情報処理システム10及びオペレーティング・シス

(4)

特開平9-330230

5

テム環境について説明する。

【0008】情報処理システム10は、高性能プロセッサ12、記憶システム14、システム・バス16、表示装置20を制御する表示サブシステム18、カーソル制御装置22、及び入出力制御装置24を備え、これらの装置がすべてシステム・バス16で接続されたグラフィック・ワークステーションなどである。図1に示したような情報処理システム10は、OS/2（OS/2は、インターナショナル・ビジネス・マシーンス・コーポレーションの登録商標）など市販の周知のマルチタスク・オペレーティング・システム26（図2に示す）によって動作する。情報処理システム10を動作させる際に、OS/2オペレーティング・システムが制御するタスクの1つは、特に、インターナショナル・ビジネス・マシーンス・コーポレーションから市販の製品の一部であるSOMobjectsに含まれるSOMオブジェクトを使用するプログラムの実行である。

【0009】本発明の方法及び製造物は、図2のブロック28のSOMobjectプログラムに組み込むことができる。

【0010】前述のように、所与のアラットフォーム用のアプリケーション・バイナリ・インターフェース（ABI）は、本質的に各社独自のものであり、複数のアラットフォームに簡単に移植することはできない。この間\*

1. somVaBufを作成する。

```
void *somVaBuf_create(char *vb, int size)
```

入力:

vb ユーザに割り振られたメモリへのポインタまたはヌル  
int ユーザ・メモリのサイズまたはヌル

出力:

somVaBufまたはヌル

【0013】この機能は、somVaBuf構造を作成し初期設定する。この構造は、ユーザ可変引数を含むこともできる。

【0014】2. va\_listに引数を追加する。

```
long somVaBuf_add(somVaBuf vb, char *arg, int type)
```

入力:

vb somVaBuf(somVaBuf\_createの結果)  
arg va\_listに追加する引数へのポインタ  
type 引数タイプ(TCKind)

出力:

成功 0

失敗 1

【0015】この機能は、渡された引数をsomVaBuf構造に含まれるva\_listに追加する。

【0016】3. somVaBufからva\_listを初期設定する。

```
void somVaBuf_get_valist(somVaBuf, va_list *ap)
```

入力:

vb somVaBuf構造へのポインタ

6

\* 題に対処するために、本発明は、変数リスト (va\_list) を操作するための、言語及びプラットフォームに中立なインターフェースを提供する新しいアプリケーション・プログラミング・インターフェース (API) を提供する。この改良型のAPIは、ライブラリ内にパッケージ化された少なくとも6つの機能を提供する。これらの機能は、可変引数リストのユーザからABIの低レベルの詳細をすべて隠蔽して、それらのユーザが、移植性の高いコードを記述できるようにする。

10 【0011】ABIに依存するものは、SOM変数バッファ (somVaBuf) データ構造内でユーザから「隠蔽」される。新しい機能は、somVaBuf構造に含まれるva\_listを操作する。これらの機能には、プラットフォーム特有のva\_listを含むsomVaBufを作成し、somVaBuf構造内のva\_listに引数を加え、somVaBuf構造からva\_listを取り出してそれをCライブラリ関数が作成したかのように使用し、somVaBufとそれに含まれるva\_listを破壊し、va\_list内の第1引数を調べて検討後に修正することができる。APIは、新しいva\_listが現れることにより、上記の機能を支援するライブラリを追加する。その利点は、ユーザ・プログラムが変化せず、したがって移植性が高いことである。

【0012】新しい機能は、次の通りである。

※ap va\_listへのポインタ

出力: なし

【0017】この機能は、somVaBuf構造内のva\_listを入力引数として渡されたva\_listに複写する。

【0018】4. somVaBufのクリーンアップ。

```
void somVaBuf_destroy(somVaBuf vb)
```

入力: vb somVaBufデータ構造へのポインタ

出力: なし

40 【0019】この機能は、somVaBuf構造及びsomVaBuf構造に含まれるva\_listに関連する割り振り済みメモリを解放する。

【0020】5. 初期設定したva\_list内の第1引数を取り出す。

```
unsigned long somValistGetTarget(va_list ap)
```

入力: ap 初期設定済みva\_list

出力: va\_list上の第1スカラー値

【0021】この機能によって、ユーザは、初期設定済みのva\_list内の第1スカラー値を調べることができる。

※50 【0022】6. 初期設定済みva\_list内の第1引数を

(5)

特開平9-330230

7

修正する。

```
void somvaListSetTarget(va_list ap, unsigned long v
al)
```

入力:

ap 初期設定済みva\_list

val va\_list内の第1スカラ値に置き換わる値

出力: なし

【0023】この機能により、ユーザはすでに初期設定済みのva\_list内の第1引数を置き換えることができる。

【0024】図3は、プラットフォーム特有のva\_listを含むsomVaBuf機構を呼び出すためのコードの実施態様を示すブロック図である。Void \*somVaBuf\_create(char \*vbuf, int size)。最初に機構は、ブロック310で、入力引数vbufがゼロかどうか判定する。ゼロでない場合は、ブロック312に進む。ゼロの場合、機構はブロック316に進み、そこでsomVaBufにスペースを割り振る。ブロック312で、システムは、入力引数sizeがゼロかどうか判定し、ゼロでない場合はブロック314に進む。sizeがゼロの場合は、システムはブロック316に進む。ブロック314において、システムは、sizeがVaBufのsizeよりも小さいかどうか判定し、小さい場合は、ブロック316に進み、小さくない場合は、システムはブロック318に進む。ブロック318において、システムは、VaBufをsomVaBufとして初期設定し、次にブロック320に進み、そこでシステムは、初期設定済みのsomVaBufを処理のために戻す。ブロック316の後、同様に、システムはブロック320に進む。

【0025】次に、図4と図5を参照すると、somVaBuf構造内のva\_listに引数を加える機能が、図4と図5のブロック図に示されている。これは、C言語でlong somVaBuf\_add(void\* vbuf, char\*arg, int type)の引数を有する。最初に、ブロック410で、システムは、somVaBuf内のva\_listに加えられる引数のタイプを判定する。図4と図5において、引数が浮動小数点演算または二倍精度演算であるfloatとdoubleの場合、システムは右側に枝分かれし、character、boolean、octet、ushort、short、long、ulong、enumerator、stringまたはpointerなど残りの引数の場合は、左側に枝分かれする。まず、floatとdoubleの場合について説明する。最初に、ブロック412で、システムは、引数のタイプが浮動小数点かどうか判定し、そうでない場合は、ブロック414に進み、幅をdoublewideの引数として定義する。そうである場合、システムは、幅を浮動小数点引数として設定する。いずれの場合も、処理が完了すると、システムは、次の図5に続く分岐Bに進む。

【0026】タイプが、floatとdouble以外の、前に述べたその他のタイプの場合、システムはブロック418～422に進み、そこで引数のタイプが、それぞれ、character、octet、boolean、short、u-short引数のどれ

8

であるかを判定する。ブロック418～422で、引数のタイプが、character、octetまたはboolean引数であると機構が判定した場合、システムはブロック428に進む。それぞれの検査後の更新を回避するため、機構は、各検査後に、ユーザ引数へのポインタでローカル変数wlongを初期設定する。検査がすべて完了すると、ブロック436でva\_listを更新することができる。タイプがshortでもu-shortでもない場合、システムはブロック432に進み、そこで引数をu-longとして確立する。この時点で、システムは、図5の点Aに進む。

【0027】分岐Aに沿って、機構はブロック434に進み、そこで機構は、va\_list内で4バイト使用可能かどうか判定し、使用可能な場合は、ブロック436に進み、そうでない場合はシステムはブロック438に進む。ブロック436において、機構は、wlongが指すユーザ引数をva\_listに入れてから主プログラムに戻る。ブロック438で、システムは、size-alignを4バイトにセットし、ブロック440～444に進んでから戻る。ブロック440で、機構は、va\_listに十分なメモリを割り振る。ブロック442で、機構は、somVaBuf情報ファイルを更新する。ブロック444で、機構は、位置合わせされた値をva\_listに入れてから戻る。

【0028】機構は、分岐Bに沿って、ブロック446で、va\_list内で8バイトが使用可能かどうかを判定する。使用可能である場合、システムはブロック448に進み、そうでない場合は、機構はブロック450に進む。ブロック448において、機構は、size-alignを8バイト幅に設定し、その後、機構は、前述のブロック440～444に進む。ブロック450において、機構は、widened引数をva\_listに入れてから戻る。

【0029】図6は、somVaBuf構造からva\_list構造を取り出す機能の流れを示すブロック図である。これは、C言語でvoid somVaBuf\_valist(void\* vbuf, va\_list\* va)として実施される。ブロック510で、機構は、somVaBufからva\_listを取り出す。

【0030】次に、図7は、somVaBuf構造とそれに含まれるva\_listを破壊する機能の実施態様を表す流れ図を示す。これは、CコードでsomVaBuf\_destroy(void\* vbuf)として実施される。最初に、ブロック610で、機構は、somVaBufにスペースが割り振られているかどうか判定し、割り振られていない場合は戻る。機構がスペースを割り振り済みである場合、システムはブロック612に進み、そこでメモリが解放される。次に、システムは戻る。図8は、va\_list内の第1引数を構成後に調べる機能のブロック図である。これは、C言語でunsigned long somva\_listGetTarget(va\_list ap)である。最初に、ブロック710で、機構が、入力引数apが指すva\_list内の第1スカラを見つける。第1スカラが見つかったら、この値をさらに処理するために戻す。

【0031】図9は、va\_list内の第1引数を構成後に

40

50

(6)

特開平9-330230

9

修正する機能のブロック図である。これは、C言語でvoid somva\_list&fctTarget(va\_list ap,unsigned long va1)として実施される。最初の段階すなわちブロック810で、第1低スカラの記憶場所を決定する。この記憶場所の決定後、その記憶場所に値を入れる。

【0032】本発明を好ましい実施形態に関して具体的に示し説明してきたが、当業者は、本発明の趣旨及び範囲から逸脱せずに形態及び細部に様々な変更を行うことができることを理解されよう。

【0033】まとめとして、本発明の構成に関して以下の事項を併示する。

【0034】(1) 1つまたは複数のプロセッサと、記憶システムと、1つまたは複数の入出力制御部と、プロセッサ、記憶システム及び入出力制御装置を接続するシステム・バスと、情報処理システムの動作を制御するようにプログラムされたオペレーティング・システムと、オペレーティング・システム・プログラムと連動して動作し、変数リストを操作するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース(API) プログラムとを含む情報処理システム。

(2) アプリケーション・プログラミング・インターフェース(API) プログラムが、さらに、ユーザの可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする上記(1)に記載のシステム。

(3) APIプログラムがさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を加える手段を含むことを特徴とする上記(2)に記載のシステム。

(4) APIプログラムがさらに、前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする上記(3)に記載のシステム。

(5) APIプログラムがさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする上記(3)に記載のシステム。

(6) 前記APIがさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す手段を含むことを特徴とする上記(3)に記載のシステム。

(7) 前記APIがさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第1引数を入れることを可能にする手段を含むことを特徴とする上記(3)に記載のシステム。

(8) 変数リストを操作して、アプリケーション・バイナリ・インターフェースを所与のプラットフォームから1つまたは複数の追加プラットフォームに移植するため

10

の言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェース(API)を提供する手段を含むコンピュータ読取り媒体。

(9) アプリケーション・プログラミング・インターフェースがさらに、ユーザ可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する手段を含むことを特徴とする上記(8)に記載のコンピュータ読取り媒体。

(10) API手段がさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を追加する手段を含むことを特徴とする上記(9)に記載のコンピュータ読取り媒体。

(11) API手段がさらに、前記オブジェクト変数バッファ構造内の変数リストを、入力引数として渡された前記変数リストに複写する手段を含むことを特徴とする上記(9)に記載のコンピュータ読取り媒体。

(12) API手段がさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する手段を含むことを特徴とする上記(8)に記載のコンピュータ読取り媒体。

(13) API手段がさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す手段を含むことを特徴とする上記(8)に記載のコンピュータ読取り媒体。

(14) API手段がさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リストに、第1引数を入れることを可能にする手段を含むことを特徴とする上記(8)に記載のコンピュータ読取り媒体。

(15) アプリケーション・バイナリ・インターフェース(ABI)を、情報処理システム上で動作する選択されたプラットフォームから1つまたは複数の追加プラットフォームに移植する方法であって、変数リストを操作して、前記ABIを前記選択されたプラットフォームから前記1つまたは複数の追加プラットフォームに移植するための言語及びプラットフォームに中立なインターフェースを提供するアプリケーション・プログラミング・インターフェースを提供する段階を含む方法。

(16) 前記アプリケーション・プログラミング・インターフェース(API)がさらに、ユーザ可変引数を含むこともある選択されたオブジェクト変数バッファ構造を作成し初期設定する段階を含むことを特徴とする上記(15)に記載のABIを移植する方法。

(17) 前記APIがさらに、前記オブジェクト変数バッファ構造の1つに含まれる変数リストに、渡された引数を追加する段階を含むことを特徴とする上記(16)に記載のABIを移植する方法。

(18) 前記APIがさらに、前記オブジェクト変数バ

50



(7)

特開平9-330250

11

バッファ構造内の変数リストを、入力引数として渡された前記変数リストに格納する段階を含むことを特徴とする上記(16)に記載のABIを移植する方法。

(19) 前記APIがさらに、前記オブジェクト変数バッファ構造と前記オブジェクト変数バッファ構造内に含まれる前記変数リストとに関連する割り振り済みメモリを消去する段階を含むことを特徴とする上記(16)に記載のABIを移植する方法。

(20) 前記APIがさらに、前記オブジェクト変数バッファ構造内の初期設定済み変数リストに記憶された第1スカラ値を取り出す段階を含むことを特徴とする上記(16)に記載のABIを移植する方法。

(21) 前記APIがさらに、ユーザが、前記オブジェクト変数バッファ構造内に記憶された初期設定済み変数リスト内に最初の引数を入れることを可能にする段階を含むことを特徴とする上記(16)に記載のABIを移植する方法。

【図面の簡単な説明】

【図1】本発明による方法を実行するためのシステムのブロック図である。

【図2】本発明を支援するオペレーティング・システム・プラットフォーム及びシステム・オブジェクト・モデル・プログラムを示すブロック図である。

【図3】somVaBuf機構を呼び出すためのコードの実施態

12

様を示すブロック図である。

【図4】somVaBuf構造内のva\_listに引数を加える機能を示す流れ図である。

【図5】somVaBuf構造内のva\_listに引数を加える機能を示す流れ図である。

【図6】somVaBuf構造からva\_list構造を取り出す機能を示す流れ図である。

【図7】somVaBuf構造とそれに含まれるva\_listを破壊する機能の実施態様を示す流れ図である。

【図8】構成後にva\_list内の第1引数を検査する機能のブロック図である。

【図9】構成後にva\_list内の第1引数を修正する機能のブロック図である。

【符号の説明】

10 情報処理システム

12 高性能プロセッサ

14 記憶システム

16 システム・バス

18 表示サブシステム

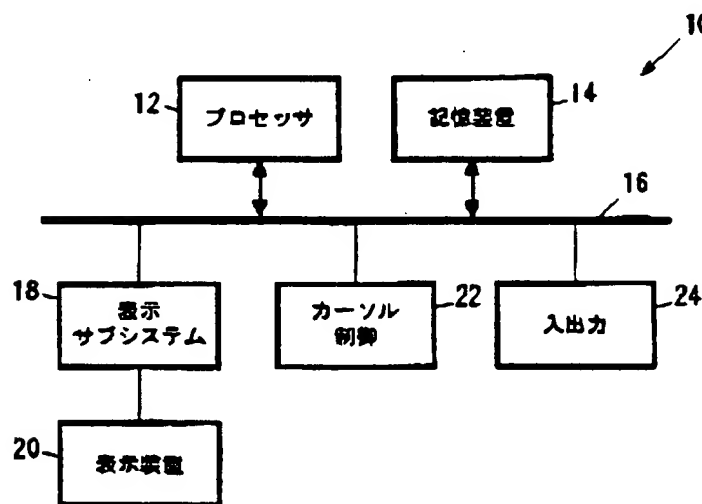
20 表示装置

22 カーソル制御装置

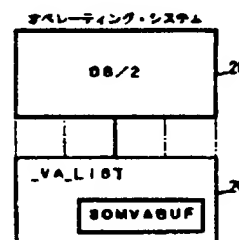
24 入出力制御装置

26 マルチタスク・オペレーティング・システム

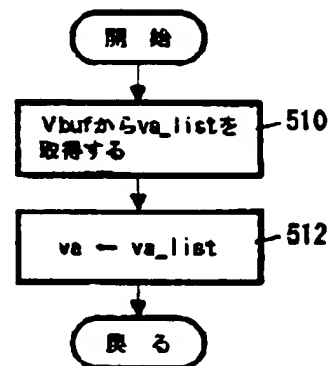
【図1】



【図2】



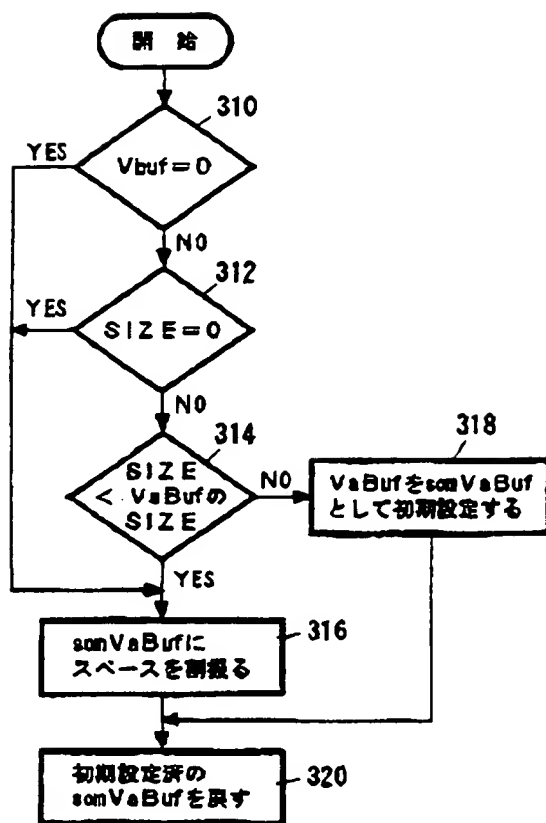
【図6】



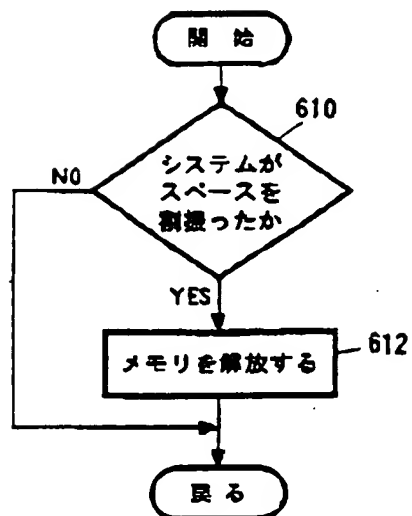
(8)

特開平9-330230

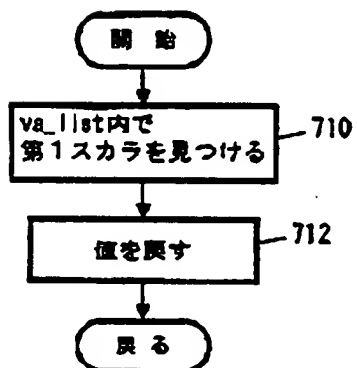
【図3】



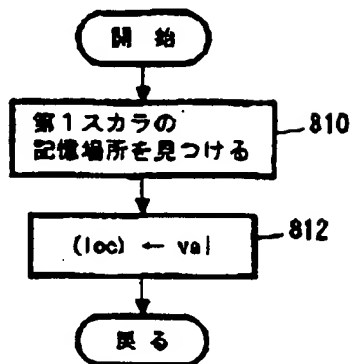
【図7】



【図8】



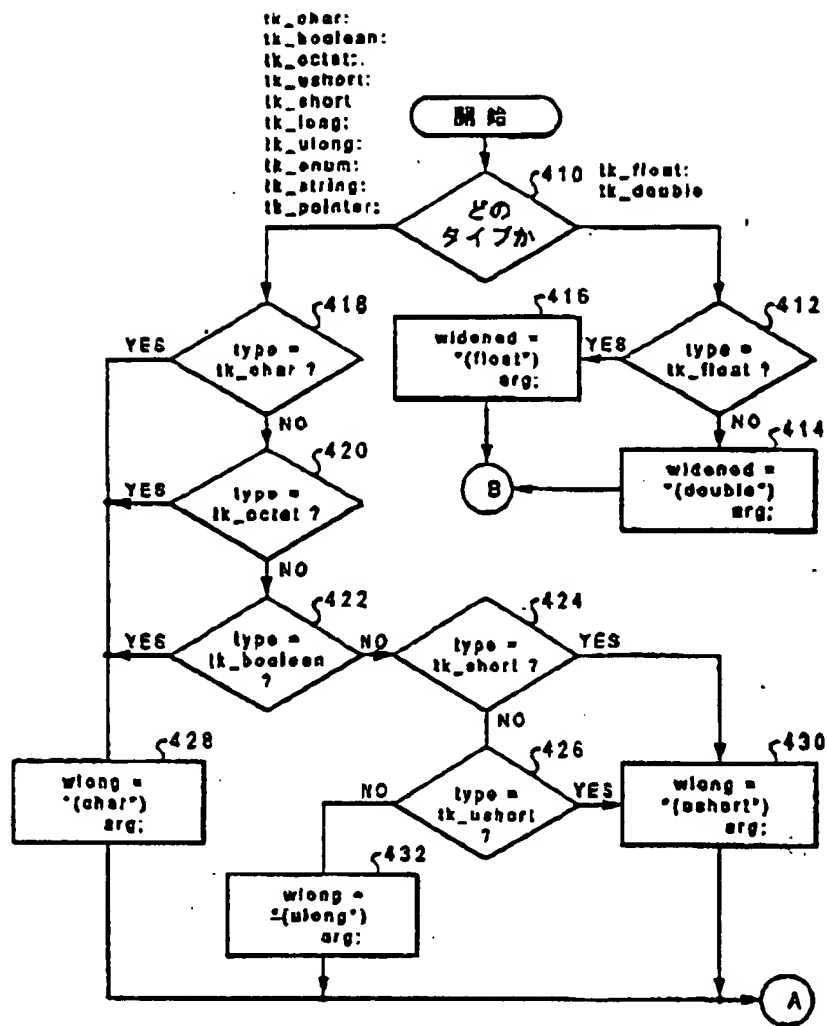
【図9】



(9)

特開平9-330230

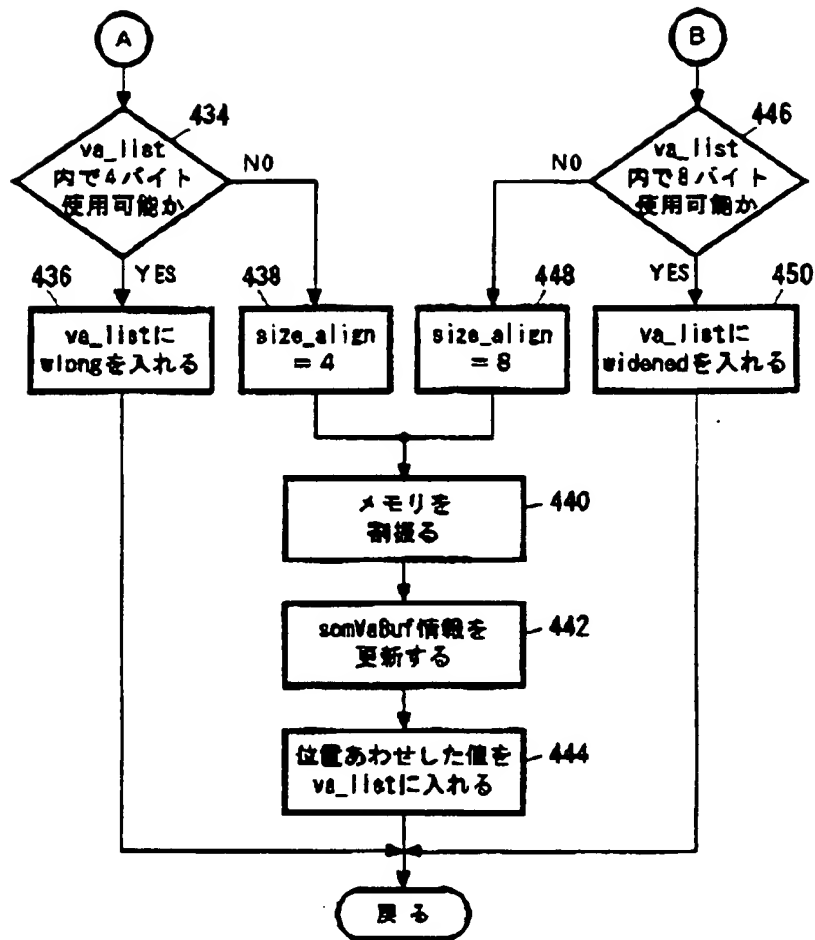
【図4】



(10)

特開平9-330230

【図5】



フロントページの続き

(72)発明者 アラカシュ・ヴィノドライ・デサイ  
アメリカ合衆国78750 テキサス州ラウン  
ド・ロック ロビン・トレール 1202